



(19)

(11) Publication number:

10198726 A

Generated Document.

PATENT ABSTRACTS OF JAPAN

(21) Application number: 09256929

(51) Intl. Cl.: G06F 17/50 H01L 21/82

(22) Application date: 22.09.97

(30) Priority: 25.09.96 US 96 719610

(43) Date of application
publication: 31.07.98(84) Designated
contracting states:

(71) Applicant: VLSI TECHNOL INC

(72) Inventor: DOCKSER KENNETH A
EHMANN GREGORY E

(74) Representative:

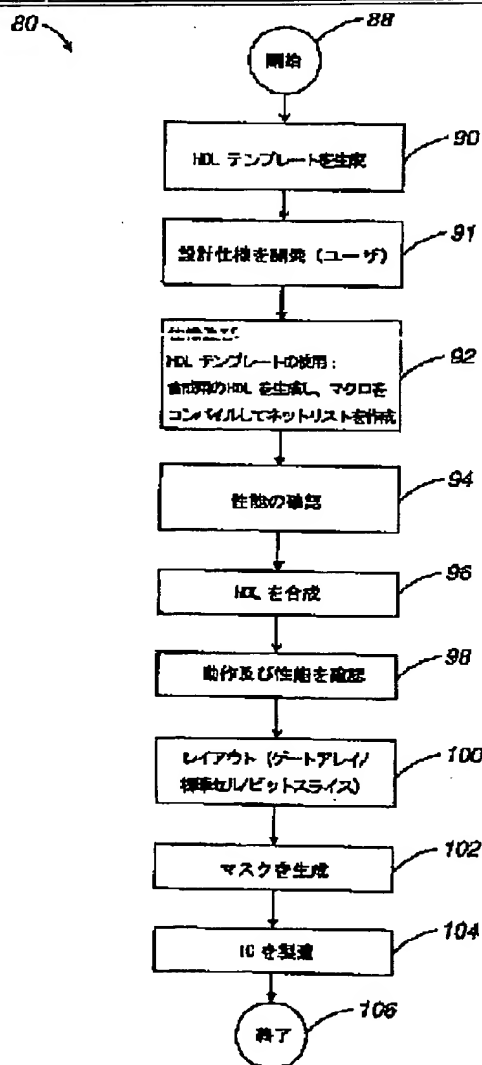
(54) METHOD AND DEVICE
FOR EFFICIENTLY MOUNTING
COMPOUND FUNCTION
BLOCK IN DESIGN OF
INTEGRATED CIRCUIT

(57) Abstract:

PROBLEM TO BE SOLVED: To protect a developer of compound function block design concerning intellectual ownership in a certain degree and to efficiently execute a compound function block.

SOLUTION: A hardware description language(HDL) template is generated (S90), the HDL templates more than one are developed by generating a parameter file and a parameter check file for the HDL template, a design specification to be used for generating the HDL for synthesization is developed (S91), the HDL for synthesization is generated and while using the design specification and the HDL template, a net list for macro blocks more than one is generated (S92).

COPYRIGHT: (C)1998,JPO



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平10-198726

(43) 公開日 平成10年(1998) 7月31日

(51) Int.Cl.⁶

識別記号

F I

G 0 6 F 17/50

G 0 6 F 15/60

6 5 4 K

H 0 1 L 21/82

H 0 1 L 21/82

C

審査請求 未請求 請求項の数20 O L (全 21 頁)

(21) 出願番号 特願平9-256929

(22) 出願日 平成9年(1997) 9月22日

(31) 優先権主張番号 08/719610

(32) 優先日 1996年9月25日

(33) 優先権主張国 米国 (US)

(71) 出願人 595173488

ヴィエルエスアイ テクノロジー インコ
ーポレイテッド

アメリカ合衆国 カリフォルニア州

95131 サン ホセ マッケイ ドライブ
1109

(72) 発明者 ケネス エー. ダックサー

アメリカ合衆国, カリフォルニア州,

サン ノゼ, テラ カタ コート 3132

(72) 発明者 グレゴリー イー. エマン

アメリカ合衆国, イリノイ州, スリー

ビー ホロウ, スーレイ レーン 825

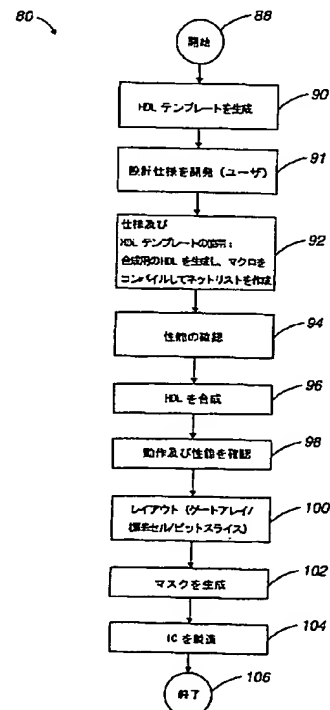
(74) 代理人 弁理士 長谷川 芳樹 (外5名)

(54) 【発明の名称】 集積回路の設計において複合機能ブロックを効率的に実装する方法及び装置

(57) 【要約】

【課題】 複合機能ブロック設計の開発者に対してある程度の知的所有権に関する保護を与えると共に、複合機能ブロックを効率的に実施すること。

【解決手段】 本発明に係る方法は、HDLテンプレートを生成し、HDLテンプレートのためのパラメタファイル及びパラメタチェックファイルを生成することにより、1つ以上のHDLテンプレートを開発し、合成用HDLの生成に際して用いるための設計仕様を開発し、合成用HDLを生成し、設計仕様、及びHDLテンプレートを使用して1つ以上のマクロブロックのためのネットリストを生成する。



【特許請求の範囲】

【請求項1】 コンパイルツールを用いて集積回路設計を開発するための方法であって、

(A) (a) ハードウェア記述言語テンプレートを生成するとともに、(b) 前記ハードウェア記述言語テンプレートのためのパラメタファイル及びパラメタチェックファイルを生成することにより、1個以上の前記ハードウェア記述言語テンプレートを開発するステップと、

(B) 合成用ハードウェア記述言語の生成に際して用いるための設計仕様を開発するステップと、

(C) 前記合成用ハードウェア記述言語を生成するステップと、

(D) 前記設計仕様及び前記ハードウェア記述言語テンプレートを使用して1個以上のマクロブロックのためのネットリストを生成するステップと、
を備える方法。

【請求項2】 前記ネットリストを合成するステップと、

合成された前記ネットリストを用いてレイアウトを生成するステップと、

を更に備える請求項1記載の方法。

【請求項3】 合成用に生成された前記ハードウェア記述言語は、前記ハードウェア記述言語で定義されていない1個以上のマクロブロックをインスタンス化する、請求項1記載の方法。

【請求項4】 前記マクロブロックのための前記ネットリストを生成する前記ステップは、前記設計仕様に基づいて前記マクロブロックを目標ライブラリに直接コンパイルするステップを含んでいる、請求項1記載の方法。

【請求項5】 前記合成用ハードウェア記述言語は、前記設計仕様に関連付けられたパスワードのユーザ入力に基づいて前記設計仕様から生成される、請求項1記載の方法。

【請求項6】 前記設計仕様を開発する前記ステップは、前記パラメタファイルに含まれる情報を使用して、ユーザからの入力を受け取るように構成されているメニューを生成するステップ、及び前記メニューを表示するステップを含んでいる、請求項1記載の方法。

【請求項7】 前記設計仕様を開発する前記ステップは、前記パラメタチェックファイルに含まれる情報を使用して前記パラメタファイルに含まれるパラメタの範囲を検査するステップを更に含んでいる、請求項6記載の方法。

【請求項8】 前記合成用ハードウェア記述言語を生成する前記ステップは、
前記ハードウェア記述言語テンプレートを暗号化するステップと、
この暗号化済ハードウェア記述言語テンプレートを解読するステップと、
この解読済ハードウェア記述言語テンプレートを実行す

るステップと、

を含んでいる、請求項1記載の方法。

【請求項9】 前記解読済ハードウェア記述言語テンプレートによって呼び出された1個以上の従属暗号化済ハードウェア記述言語テンプレートを解読するステップと、

前記解読済従属ハードウェア記述言語テンプレートを実行するステップと、

を更に備えている請求項8記載の方法。

10 **【請求項10】** 前記集積回路設計に用いる1個以上のマスクを生成するステップを更に備える請求項1記載の方法。

【請求項11】 請求項10に記載の方法によって形成される集積回路。

【請求項12】 コンパイルツールを用いて集積回路を設計する方法であって、

(A) (a) 異なる機能を提供するために複数のハードウェア記述言語テンプレートを生成するとともに、

20 (b) 前記複数のハードウェア記述言語テンプレートのための複数のパラメタファイル及び複数のパラメタチェックファイルを生成することにより、前記複数のハードウェア記述言語テンプレートを開発するステップと、

(B) 合成用ハードウェア記述言語の生成に際して用いるための設計仕様を開発するステップと、

(C) 前記合成用ハードウェア記述言語を生成するステップと、

30 (D) 前記設計仕様及び前記複数のハードウェア記述言語テンプレートを使用して、1個以上のマクロブロックのためのネットリストを生成するステップと、
を備える方法。

【請求項13】 前記複数のハードウェア記述言語テンプレートは、1個以上の下位テンプレート、及び、その下位テンプレートを呼び出すように構成されている1つ以上の上位テンプレートを備えている、請求項12記載の方法。

【請求項14】 前記複数のパラメタファイルは、前記上位テンプレートに関連する第1のパラメタ、及び前記下位テンプレートに関連する第2のパラメタを備えている、請求項13記載の方法。

40 **【請求項15】** 前記合成用ハードウェア記述言語を生成する前記ステップは、
前記上位テンプレートを暗号化するステップと、
前記下位テンプレートを暗号化するステップと、
前記上位テンプレートを解読するステップと、
前記下位テンプレートを解読するステップと、
前記下位テンプレートを実行するように構成されている前記上位テンプレートを実行するステップと、
を更に含んでいる、請求項13記載の方法。

【請求項16】 中央演算装置、及びこの中央演算装置に接続されている記憶装置を有するコンピュータシステ

ムと、
集積回路設計用の設計仕様を開発する際に用いるため
の、1個以上のハードウェア記述言語テンプレート、1
個以上のパラメタファイル、及び1個以上のパラメタチ
ェックファイルを取得する手段と、
前記設計仕様を開発する手段と、
前記設計仕様及び前記ハードウェア記述言語テンプレ
ートを使用して合成用ハードウェア記述言語を生成する手
段と、
前記設計仕様及び前記ハードウェア記述言語テンプレ
ートを使用してネットリストを生成する手段と、
を備える集積回路設計ツール。

【請求項17】 前記設計仕様開発手段は、前記パラメ
タファイルに含まれる情報を使用して、ユーザから入力
を受け取るように構成されているメニューを生成し、こ
のメニューを表示する手段を含んでいる、請求項16記
載の集積回路設計ツール。

【請求項18】 前記設計仕様開発手段は、前記パラメ
タチェックファイルに含まれる情報を使用して前記パラ
メタファイルに含まれるパラメタの範囲を検査する手段
を更に含んでいる、請求項17記載の集積回路設計ツ
ール。

【請求項19】 前記設計仕様開発手段は、前記設計仕
様を満足させるために用いる適当なマクロを決定する手
段を更に含んでいる、請求項18記載の集積回路設計ツ
ール。

【請求項20】 前記マクロ決定手段は、前記パラメタ
ファイルを用いて所望のオプションセットを複数生成す
る手段を含んでいる、請求項19記載の集積回路設計ツ
ール。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、集積回路の設計に
関し、特に、コンパイル (compilation) 及び合成 (sy
nthesis) による集積回路の自動化設計に関する。

【0002】

【従来の技術】 過去数十年にわたり、集積回路 (IC)
は、近代電子装置に必要不可欠なものとなってきた。

「既製」のコンポーネントを使用するよりも、カスタム
ICまたはセミカスタムICを使用することがしばしば
望ましい。簡単な論理設計のカスタムICまたはセミカ
スタムICにより、特定の性能制約または回路要求を満
たし、回路の製造を支援できるように、設計用及び製造
用の自動化システムが開発されている。自動化設計シス
テムはまた、ライフサイクルの短いICや非常に複雑な
ICの設計に広く用いられている。

【0003】 計算機援用設計 (CAD) システムの一種
として、「論理合成システム」が知られている。このよ
うなシステムでは、入力、出力及びハイレベル設計記述
がハードウェア記述言語 (HDL) を用いてコンピュー

タに入力される。その後、合成ソフトウェアは、HDL
で記述された機能を実行する論理設計を生成する。

【0004】 合成を使用する一般的な集積回路設計およ
び製造プロセスは、通常、レジスタトランスファレベル
(RTL) のハイレベル設計記述を用いて開始される。
RTL記述は、例えば、市販のVHDLまたはVerilog-
HDLといったHDLを用いて生成される。このHDL
を合成されることで、HDLを用いて記述された機能
を実行するコンポーネント及びこれらのコンポーネントの
相互接続を特定する「ネットリスト」(すなわち、「ネ
ット」のリスト) を生成することができる。しかしなが
ら、現実のウェハ上におけるコンポーネントの配置設
計、及び配線接続のトポグラフィ (topography) は、続
く「レイアウト」段階のために残される。

【0005】 従来の集積回路の製造方法10の概略が、
図1中にステップ12から始まるフローチャートによ
って記載されている。ステップ14では、一組の設計仕様
が開発される。一般的に、これらの仕様には、集積回路
全体の性能、及び特定の機能、寸法、及びチップ上にお
けるコンポーネントの配置特徴が含まれていても良い。

【0006】 論理設計者は、HDLを用いてステップ1
6にて設計される論理のRTL記述を生成する。共通ハ
ードウェア記述言語には、前述のVHDL及びVerilog-
HDLが含まれるが、他の好適な言語 (例えば、開発者
が知的所有権を保持するHDL) も使用することができ
る。この後、設計のRTL記述は、ステップ18にてネ
ットリストを合成するために用いられる。ネットリスト
は、ハードウェア記述言語 (HDL) によっても記述さ
れ得る。ステップ20は、ネットリストの動作 (behavi
or) 及び機能 (functionality) を確認し、動作及び機
能が仕様を満足しない場合には、再度、ステップ16及
びステップ18を実行させる。

【0007】 前述のように、ネットリストは、接続され
るコンポーネント (「セル」あるいは「モジュール」と
して知られている) を特定するが、正確な配線形状 (wi
ringtopography) は特定しない。セルは、他のセルのピ
ンと相互接続するためのピンを1本以上有している。

「ネットリスト」は、セルのピン間の接続性を定義する
「ネット」を含んでいる。換言すれば、「ネット」は、
多数のセルの電氣的に等価なピンのセットであり、これ
らのピンは、共通電気ノードを形成するために相互に接
続されなければならない。ネットリストにより記述され
ているコンポーネント、あるいは、セルは、設計仕様を
満たす論理設計を形成する。

【0008】 さらに、図1の従来のプロセスを参照する
と、ステップ24にて、論理設計者は、確認済のネット
リスト記述をレイアウトツール中に転送する。レイアウト
ツールにより実行されるレイアウトステップ24は、
集積回路ダイまたはチップの「レイアウトエリア」上
におけるセルの実際の物理的配置を決定して、ゲートアレ

イ、あるいは、標準セルを形成する。「レイアウトエリア」は、ICの能動部品(active component)のために割り当てられる領域である。レイアウトプロセスの「配置」ステップは、数十万のゲートを有するIC用コンピュータワークステーション上で計算に数日を要する、非常に時間のかかるステップである。セルのピン間の接続または「ワイヤ」の実際の経路も、レイアウトステップ24にて決定される。

【0009】ステップ24で生成されたカルテック中間フォーマット(C.I.F.)データは、ステップ26において、集積回路マスクが生成されるマスク製造ツールに転送される。このマスクは、集積回路チップ、または、ダイを生成するために用いられる。このマスクは、C.I.F.データを読み取るために備えられているマシン上で生成される。このC.I.F.データは、ハードディスク、磁気テープ、フロッピーディスク、あるいは、他の伝達媒体を通じてこのマシンに転送することができる。マスク生成マシンは、ネットリストを合成するマシンの一部、あるいは、同一のマシンであっても良い。

【0010】ステップ28では、集積回路が製造される。従来の回路製造方法は、フォトリソグラフィプロセスにおいて、ステップ26で生成されたマスクを使用する。チップ自体が製造されてしまうと、ダイ上の集積回路は、外部回路に対する接続を備えなければならない。これは、一般的に、集積回路にボンディングワイヤ及び/またはリードフレームを取り付けることによって達成される。この後、回路は、プラスチック等のパッケージ材料に封入される。集積回路の設計及び製造は、30で示されているこの時点で完了する。

【0011】集積回路がより複雑になるにつれて、論理設計者には、複合機能ブロックを集積回路設計中に迅速に実装する能力を備えていることが要求される。集積回路設計中に複合機能ブロックを迅速に実装する1つの手法としては、固定ネットリストの論理回路へのインスタンス化(instantiation)が挙げられる。論理設計への固定ネットリストのインスタンス化は、マイクロプロセッサ等の固定定義装置用のネットリストに対しては上手く機能するが、様々な構成を備え得るマルチピラー等のブロック用の固定ネットリストのインスタンス化は非効率的である。例えば、設計が14ビットマルチピラーブロックを要求するが、24ビットの固定ネットリストマルチピラーしか利用できない場合、得られる設計は、結局、要求されるものよりも大きく、遅いものとなる。

【0012】集積回路設計中に複合機能ブロックを迅速に実装する他の手法としては、マクロテンプレートを作成ツールに対して供給する前に、設計者によるマクロテンプレート中のパラメタの変更を可能にするRTLで生成されているマクロテンプレートを使用することが挙げられる。マクロテンプレートの合成は、目標ライブラリと詳細には関連付けられないので、この手法により得ら

れる結果は、複合機能ブロックの最適な実装とは言えない。これは、従来の論理合成ツールは、機能を原始コンポーネントまで分解した後、それらを目標ライブラリにバックアップしようと試み、しばしば特定の最適化セルを失うためである。さらに、マクロテンプレートは一般的にHDLコードの羅列なので、マクロテンプレートが顧客に配布される際、マクロテンプレートの開発者は、マクロテンプレートに関連する大量の知的所有権を開示することになる。

【0013】

【発明が解決しようとする課題】本発明は、上記した従来技術の問題点を解決するためになされたものであり、複合機能ブロック設計の開発者に対してある程度の知的所有権保護を与えると共に、複合機能ブロックを効率的に実装することを目的とする。

【0014】

【課題を解決するための手段】上記目的を達成するため、請求項1に記載の発明に係るコンパイルツールを用いた集積回路設計の開発方法は、(A)(a)ハードウェア記述言語テンプレートを作成するとともに、(b)ハードウェア記述言語テンプレートのためのパラメタファイル及びパラメタチェックファイルを作成することにより、1個以上のハードウェア記述言語テンプレートを開発するステップと、(D)合成用ハードウェア記述言語の生成に際して用いるための設計仕様を開発するステップと、(C)合成用ハードウェア記述言語を作成するステップと、(D)設計仕様及びハードウェア記述言語テンプレートを使用して1個以上のマクロブロックのためのネットリストを作成するステップと、を備えている。

【0015】また、請求項12に記載の発明に係るコンパイルツールを用いた集積回路設計の開発方法は、

(A)(a)異なる機能を提供するために複数のハードウェア記述言語テンプレートを作成するとともに、

(b)複数のハードウェア記述言語テンプレートのための複数のパラメタファイル及び複数のパラメタチェックファイルを作成することにより、複数のハードウェア記述言語テンプレートを開発するステップと、(B)合成用ハードウェア記述言語の生成に際して用いるための設計仕様を開発するステップと、(C)合成用ハードウェア記述言語を作成するステップと、(D)設計仕様及び複数のハードウェア記述言語テンプレートを使用して、1個以上のマクロブロックのためのネットリストを作成するステップと、を備えている。

【0016】さらに、請求項16に記載の発明に係る集積回路設計ツールは、中央演算装置及びこの中央演算装置に接続されている記憶装置を有するコンピュータシステムと、集積回路設計用の設計仕様を開発する際に用いるための、1個以上のハードウェア記述言語テンプレート、1個以上のパラメタファイル、及び1個以上のパラ

メタチェックファイルを取得する手段と、設計仕様を開発する手段と、設計仕様及びハードウェア記述言語テンプレートを使用して合成用ハードウェア記述言語を生成する手段と、設計仕様及びハードウェア記述言語テンプレートを使用してネットリストを生成する手段と、を備えている。

【0017】本発明は、テンプレート設計者の知的所有権に対してある程度の保護を与えると同時に、カスタム設計を作成するために設計に関するパラメタを論理設計者が修正できるようにする暗号化済ハードウェア記述言語（HDL）テンプレートを論理設計者に提供する改良方法を提供する。本発明の主たる利点は、暗号化済HDLテンプレート及び関連マクロを論理設計者からの入力と有効に結びつけてカスタム出力を提供することができることである。カスタム出力は全てのパラメタを含んではいないので、論理設計者には、特定の実装のためのHDLだけが提供され、HDLテンプレートに関連する知的所有権の全てが論理設計者に対して明かされることはない。これにより、テンプレート設計者に対して知的所有権に対するある程度の保護を与えると同時に、論理設計者は、複合マクロブロックを効率的に実装することができるようになる。

【0018】本発明の他の利点は、所定の設計に用いるための最適なマクロが、論理設計者によって提供される入力を用いてHDLテンプレート中にインスタンス化された各マクロをコンパイルすることにより生成されることにある。このことは、設計仕様の達成に役立つばかりでなく、論理設計の品質の向上にも役立つ。さらに、従来の合成ツールとは異なり、マクロブロックコンパイラは、目標ライブラリの詳細な情報を用いて機能する。このコンパイラは、適当な複合セルを小さな領域に直接インスタンス化し、得られるチップの経路決定性（routability）を向上させる。

【0019】

【発明の実施の形態】以下、本発明に係る集積回路設計中に複合機能ブロックを効率的に実装するための方法および装置を具体化した発明の実施形態について図面を参照して説明する。

【0020】図2は、本発明に係る集積回路の製造プロセスを示している。このプロセス80は、ステップ90で開始する。ステップ90では、ハードウェア記述言語（HDL）のテンプレートが生成される。HDLテンプレートは、ユーザあるいは論理設計者が「制御」あるいは修正しうる、設定可能なHDLコードの一部である。HDLテンプレートの生成方法は、図4を参照して後述する。HDLテンプレートが生成された後、ステップ91では、ユーザによって提供された情報を用いて設計の仕様が開発される。一般的に、設計仕様の開発ステップは、図1を参照して上述したように、集積回路全体の能力、また、チップ上における特定の寸法、及び特定のコンポーネントの配置に関連するパラメタを含んでいても良い。ユーザにより提供された情報を使用する設計仕様の開発は、図5を参照して後述する。

ンポーネントの配置に関連するパラメタを含んでいても良い。ユーザにより提供された情報を使用する設計仕様の開発は、図5を参照して後述する。

【0021】ステップ92では、ステップ91にて開発された設計仕様をステップ90にて生成されたHDLテンプレートと共に用いて、集積回路用ネットリストを合成するために用いることの可能なマクロネットリスト及びHDLが生成される。パラメタ及びパラメタチェックファイルもまた、合成可能なHDLの生成の際に用いることができる。合成可能なHDL、すなわち「合成用HDL」、の生成プロセスに含まれる詳細なステップは、図8を参照して後述する。この後、ステップ94にて、マクロネットリストの性能が確認される。コンパイル済マクロの性能が仕様を満足しない場合には、新規のコンパイル済マクロを生成しても良い。すなわち、ステップ92は、好適なマクロネットリストが取得されるまで繰り返すことができる。

【0022】HDLの生成を可能にするために、特定のパラメタの組み合わせに固有のパスワードを要求しても良い。このようなパスワードは、特定のホストコンピュータに関連付けられていても良く、また、有効期限を有していても良い。パスワードを使用することにより、ベンダーは、特定の構成についてだけ合成可能HDLを生成するようユーザを規制することができる。したがって、ユーザは、全ての順列をコンパイルすることによってテンプレートの内容を取得することが不可能となる。

【0023】マクロネットリストが仕様を満足する場合、ステップ96にて、合成用HDLからネットリストが合成される。このネットリストは、HDLで記述することができ、任意の好適な合成ソフトウェアを使用することによって合成することができる。ステップ98は、合成されたネットリストの動作の確認である。合成済みのネットリスト及びマクロネットリストの動作または性能が仕様を満足しない場合には、ステップ98は、ステップ92、94及び96を繰り返し実行させる。合成ネットリストの動作及び性能が確認されると、ステップ100にて論理設計者は、確認済ネットリストの記述をレイアウトツール中に転送する。レイアウトツールによって実行されるレイアウトステップ100は、ゲートアレイ、標準セル、あるいは、ビットスライスを形成するために、集積回路ダイ又はチップの「レイアウトエリア」上におけるセルの実際の物理的配置を決定する。セルのピンの間の接続、あるいは、「ワイヤ」の現実の経路もまた、レイアウトステップ100にて決定される。

【0024】ステップ100にて生成されたデータは、一般的に、カルテック中間フォーマット（C.I.F.）であるが、このデータは、ステップ102にて、集積回路のマスクが製造されるマスク製造ツールへ転送される。このマスクは、集積回路チップ、あるいは、ダイを生成するために用いられる。マスクの生成は、図1を参照して

説明済みである。

【0025】マスクが生成された後、ステップ104にて集積回路が製造される。従来の回路製造方法は、フォトリソグラフィプロセスにおいて、ステップ102で生成されたマスクを使用する。チップ自体が製造されてしまうと、ダイ上の集積回路は、外部回路に対する接続を備えなければならない。これは、一般的に、集積回路にボンディングワイヤ及び／またはリードフレームを取り付けることによって達成される。この後、この回路は、プラスチック等のパッケージ材料に封入される。集積回路の設計及び製造は、106で示されているこの時点で完了する。

【0026】図3には、集積回路製造システム31のブロック図が図示されている。システム31は、中央演算装置(CPU)32、I/Oスロット34、キーボード36、モニタ38、ROM40、RAM42、ディスクドライブ装置44、マスク生成装置46及びIC製造装置48を備えている。CPU32はI/Oポート34、及び、ユーザ入力装置、例えば、キーボード36と接続されている。HDLはI/Oポート34、ユーザ入力装置36、あるいは、他の入力チャネル、例えば、ディスクドライブ44、を介して受け取ることができ、システム31に入力することができる。

【0027】一般的に、I/Oポート34を介して受け取られるHDL、あるいは、HDLテンプレートは、他のマシン(コンピュータ)から送られる。これは、例えば、ネットリストが他のコンピュータによって合成されるときに事例である。ユーザ入力装置36は、通常、キーボードの形式を採る。これにより、ユーザ、おそらく、論理設計者は、設計仕様を入力し、あるいは、CPUによって実行されるネットリストマルチピラーを制御することができる。一般的に、入力装置、または、キーボード36を使用する論理設計者は、CPU32に接続されているモニタ38を使用する。

【0028】システムプロセッサ31には、種々の型式のデジタル記憶容量が設けられていることが好ましい。図3に示されるように、一般的に、このデジタル記憶装置には、ROM40、RAM42、及びディスクドライブ44が含まれる。ディスクドライブ44は、I/Oポート34又はユーザ入力装置36から受け取ったHDLを記憶し、あるいは、HDLをシステムに入力するために用いることができ、また、ハードウェア記述プロセッサ31及びそのCPU32上で実行中のプロセスによって生成されるマスク生成データを記憶することができる。ディスクドライブ44は、他の永久記憶装置、例えば磁気テープやフロッピーディスク、によって置換あるいは増補することができる。既述のように、元のネットリストは、例えば、I/Oポート34又はユーザ入力装置36のいずれか一方を通じて入力し、あるいは、システム31上で直接合成することができる。

【0029】システム31は、ネットリストからマスク生成データを開発する。このマスク生成データは、例えば、ディスクドライブ44のようなデジタル記憶装置に格納される。マスク生成装置46は、マスク生成データをCPU32から受け取る。この他に、(図示しないが)マスク生成装置46は、マスク生成データを、ディスクドライブ44等のデジタル記憶装置から直接受け取っても良い。マスク生成装置46は、ハードウェア記述プロセッサの一部であっても良いし、別の装置であっても良い。このマスク生成データ、あるいは、C.I.F.は、フォトリソグラフィマスクを生成するためにマスク生成装置46によって用いられる。これらのマスクは、集積回路のコンポーネントをウェハ上に形成するために、集積回路製造装置48によって使用される。このマスクは、集積回路上のコンポーネントやこれらのコンポーネント間の接続を形成するために十分なものである。集積回路製造装置48は、当業者に周知の半導体製造設備、例えばエッチング装置、化学的気相成長(CVD)機、リソグラフィ機等、を有している。

【0030】システム31によるプロセスの最終生成物は、パッケージされた集積回路50である。このパッケージIC50は、マスク生成装置46によって生成されたマスクを使用して生成されたダイ52を含んでいる。この半導体ダイ52は、一般的に、デジタル集積回路、及び回路をリード53に接続するためのI/Oパッド51を有している。I/Oパッド51は、任意の従来方法、例えば、ボンディングワイヤによってリード53と接続することができる。

【0031】既述のように、ネットリストは、コンポーネント(セル)及びそれらの理論的相互接続を順に特定するネットを記述する。換言すれば、ネットリストは、論理設計の理論的な記述である。複合論理設計に対しては、テンプレートをを用いることで、RTLを合成ツールに入力する前に論理設計者が論理設計に関連するパラメタを修正できるようにすることができる。

【0032】図4は、HDLテンプレート生成プロセスを詳細に示す。このプロセス90は、300にて始まり、ステップ302では、種々の機能に固有の多数のHDLテンプレートが生成される。一つの実施形態では、これらの種々の機能はハイレベル機能であっても良い。ステップ304では、パラメタファイル及びパラメタチェックファイルが各HDLテンプレートについて生成される。各HDLテンプレートのパラメタに関連する部分は、HDLテンプレートから取得され、パラメタファイルを生成するために用いられる。このため、パラメタファイルは、パラメタファイルに関連付けられたHDLテンプレートに対するグローバルパラメタを定義するために用いられる情報を含んでいる。

【0033】図5を参照して後述するように、このパラメタファイルは、通常、HDLテンプレート用のユーザ

入力ウィンドを定義するために用いることのできる情報も含んでいる。すなわち、このパラメタファイルは、通常、グラフィカルユーザインターフェイス（GUI）を定義するために用いられる。一般的に、コンパイルツールに固有のスクリプト言語で記述されるパラメタチェックファイルは、パラメタファイル内のパラメタを検査してパラメタが正当なエントリであるか否かを判定するために用いられる。換言すれば、パラメタチェックファイルは、パラメタファイル上にて誤り検査を行うために用いられる。パラメタファイル及びパラメタチェックファイルの例は、図10、並びに図11及び図12にそれぞれ示されている。

【0034】ステップ306では、HDLテンプレートが暗号化される。テンプレートを暗号化するために用いることのできるサードパーティ製暗号化及び解読ソフトウェアツールは、「FLEXcrypt」の製造元であるカリフォルニア州キャンベル市のグローブ Trotter ソフトウェア（Globetrotter Software, Inc.）を含む多くのベンダーから入手可能である。HDLテンプレート及びマクロの生成及び暗号化のプロセスは、308にて完了する。

【0035】図5は、設計仕様開発プロセスを詳細に示す。このプロセス91は400にて始まり、ステップ402では、パラメタファイルを用いてメニューが生成され、スクリーンディスプレイ上に表示される。パラメタファイル内のメニュー関連情報には、例えば、ディスプレイ中のメニュー項目のタイトルの定義に用いることのできる可能な「メニュータイトル」が含まれていても良い。表示されるメニューの内容は、図6を参照して後述する。メニューが生成されて表示された後、ステップ404にてユーザからの入力を受け取る。この入力には、パラメタ形式の設計仕様や、パラメタを使用するオペレーションの実行要求や、設計仕様開発プロセスの中断要求が含まれる。

【0036】ステップ406では、ユーザから受け取った入力のタイプに関する判定がなされる。この実施形態では、パラメタ、ユーザ入力受け取りプロセスの中断要求、あるいは、メニュー生成に用いられたパラメタファイルに関連するパラメタのコンパイルまたは評価、のいずれのタイプに入力が当てはまるか判定される。入力タイプがパラメタであると判定された場合、プロセス制御は、ステップ404に戻る。入力タイプがユーザ入力受け取りプロセスの中断要求であると判定された場合、本プロセスは422で完了する。

【0037】ステップ406にて、入力はパラメタファイル中のパラメタのコンパイルまたは評価要求であると判定された場合には、プロセス制御は、ステップ408に進む。ステップ408では、適当なパラメタチェックファイルを用いてパラメタ範囲が検査される。すなわち、このパラメタチェックファイルは、パラメタチェッ

クファイルに関連付けられたパラメタファイル中の入力済パラメタ上で誤り検査を実行する。この誤り検査の後、ステップ410では、入力済パラメタのいずれかが範囲を外れていないかどうか判定される。1個以上のパラメタが範囲を外れていると判定された場合には、ステップ412にてスクリーン上にエラーメッセージが表示される。エラーメッセージが表示された後、プロセス制御は、ユーザに範囲外のパラメタを変更する機会を与えるために、ユーザからの入力を受け取るステップ404に戻る。

【0038】ステップ410にて、範囲外のパラメタが1個もないと判定された場合には、プロセス制御は、ステップ414に進む。ステップ414では、ステップ404にてユーザから受け取った入力が、HDLテンプレートを評価し、結果として得られる設計に用いられるマクロブロックの適切なバージョンを決定することの要求であったか、あるいは、コンパイル要求であったかが判定される。入力が設計の評価要求であった場合には、ステップ416にて、パラメタに関連付けられたHDLテンプレートが解読ツールを用いて解読される。HDLテンプレートが解読されると、現実の設計において用いるための、最良の、あるいは、最適なマクロブロックアーキテクチャがステップ418にて決定される。コンパイラツールは、要求されたマクロブロックの異なるアーキテクチャについてのネットリストを生成し、最良のものを選択する。例えば、マクロブロックが加算器で用いられる場合には、といった、加算器の幾つかの異なるアーキテクチャ、例えばキャリルックアヘッド（carry-look ahead）やキャリスキップ（carry skip）、がコンパイルされ、得られるネットリストのサイズ、タイミング、及び他の属性が解析されることになる。最良のマクロブロックは、例えばタイミングや性能といった点において特定の設計仕様を満足するのに最も役立つマクロブロックである。最適マクロブロックのアーキテクチャが識別された後、プロセス制御は、ステップ404に戻る。ステップ412にて、入力はコンパイル要求、すなわち、コンパイル命令、であると判定された場合には、設計仕様の開発プロセスは、420で完了する。

【0039】図6は、パラメタファイル中に設けられた情報から生成されるメニュー表示を図示する。メニュー500は、一般的に、GUIであり、ユーザがキーボード、及び好適なポインティング装置、例えばマウス、を用いてメニュー500のフィールドにアクセスできるようにモニタ上に表示される。図示の実施形態では、メニュー500は、同期先入れ先出し（FIFO）メモリに使用される入力情報を取得するように設定される。メニュー500は、ユーザにより与えられる情報の全タイトル、すなわち「同期FIFOテンプレートコンパイラ」を定義するために用いられるメニュータイトル502、を有している。

【0040】FIFOと連携するランダムアクセスメモリ(RAM)に関する種々の物理パラメタは、ユーザが与えることができる。例えば、「ワード奥行(word depth)」504及び「ワード幅(word width)」506を特定することが可能である。ワード奥行504及びワード幅506は共に、それぞれ、RAMについての奥行及び幅の表示を与える整数である。「RAMタイプ」508は、FIFOのメモリコアについての構成を表示する。図示の実施形態では、ユーザは、論理設計の要件に応じて、RAMタイプ508をラッチタイプ510、フリップ／フロップタイプ512、あるいは、RA5タイプ514の中から選択する。

【0041】FIFOのメモリコアが満杯であるか否かを示すために用いられる「フルフラグ」516信号の追加は、ユーザが選択することができる。同様に、メモリコアが空であるか否かを示す「エンプティフラグ」518も選択することができる。FIFOのうち何ワードがデータを含んでいるかを示す「FIFO奥行インジケータバス」520も選択することが可能である。

【0042】ラッチ510がRAMタイプ508として選択される場合、論理設計者が望むなら、テストモードピンを有するラッチを具体化、あるいはインスタンス化することができる。「ラッチRAM用スキャンテストモードピン」選択522は、入力選択ボックス524から適当なスキャンオプションを選択することにより実行することができる。入力選択ボックス524中のスキャンオプションには、図示のようにRAMに対して用いられるラッチはスキャン型であるべきでないことを示す「なし」オプションが含まれ得る。他のオプションには、RAMに対して用いられるラッチはテストモードピンに対するバッファ信号(buffered signal)を含むことを示す「バッファ」オプション、及びあらゆる種類のバッファリングを用いないでラッチの全テストモードピンが互いに接続されるべきであることを示す「非バッファ」オプションが含まれる。

【0043】「コンポーネント名」526は、設計されているコンポーネントの名称を特定するために用いられる。ある実施形態では、デフォルトのコンポーネント名528は、メニュー500によって特定されるコンポーネントの名称であっても良い。例えば、図示するように、デフォルトのコンポーネント名528は、「fifo」である。同様に、「出力ファイル名」530は、コンポーネント名526によって識別されたコンポーネントを含むファイル名を特定するために用いられる。デフォルト出力ファイル名532は、コンポーネントの名称と同じにすることができ、図示の実施形態では「fifo」である。

【0044】メニュー500はまた、コマンドを選択して呼び出すために活性化されるインターフェースを有している。「評価」コマンド534は、図5を参照して前

述したように、入力済パラメタの範囲を検査したり、入力済パラメタを用いて生成されたテンプレートを解読したり、テンプレート中に与えられた設計仕様を満足するために用いる最良のマクロブロックアーキテクチャを決定するために選択することができる。「コンパイル」コマンド536は、図5を参照して前述したように、入力済パラメタの範囲を検査したり、テンプレートを解読したり、入力済パラメタを使用して合成用HDLを生成したり、マクロブロックをコンパイルするために選択することができる。「閉じる」コマンド538は、メニュー500を用いた情報入力プロセスを中止または終了させるために選択することができる。ある実施形態では、「ヘルプ」コマンド540が選択されると、メニュー500中のいくつかのパラメタを定義することが可能な情報を含む「ヘルプメニュー」が表示されるようにすることができる。他の実施形態では、ヘルプコマンド540は、メニュー500中に図示されるパラメタに関連するヘルプメニューを含み、特定のヘルプメニューにアクセスするために用いることが可能な情報を含む総合メニューを表示しても良い。

【0045】図7は、設計仕様を開発する際に用いる適当なマクロブロックアーキテクチャを決定するプロセスを詳細に示している。このプロセス418は、600にて始まり、ステップ602では、所望のオプションセットのリストを生成するため、あるいは、所望のパラメタ組み合わせのリストを生成するために、HDLテンプレートに関連付けられたパラメタファイルが用いられる。ある実施形態では、パラメタファイルをトラバースすることで、全ての可能性あるパラメタ組み合わせの順列(permutation)が見つかる。これらの順列は、この後、所望のオプションセットのリストを生成するために用いられる。他の実施形態では、所望のオプションセットのリストは、全ての可能性あるオプションセットのサブセットである。ステップ604では、オプションセットのリストから1個のオプションセットが選択される。ステップ606では、未だ選択及び解析のされていないオプションセットが残っているかどうか判断される。選択すべきオプションセットが残っていない場合、論理設計の合成に用いるための適当なマクロブロックの決定のために所望のオプションセットの全てが既に選択及び解析された旨が表示され、適切なマクロブロックを選択するプロセスは612にて完了する。

【0046】ステップ606において、選択可能な他の所望のオプションセットが残っていると判断された場合、プロセスフローは、ステップ608に進む。ステップ608では、HDLテンプレートにインスタンス化されたマクロブロックが選択オプションセットを用いてコンパイルされる。HDLテンプレートを用いてインスタンス化された各マクロブロックは、選択されたオプションセットと共同して各マクロブロックの性能を判定する

ために呼び出され、コンパイルされる。

【0047】ステップ610では、コンパイル済マクロブロックが解析される。そして、解析の結果は、通常、論理設計者が読むことのできるレポートの中で評価および提示される。このレポートには、オプションセットを用いてコンパイルされた各マクロブロックの論理設計のサイズ、速度、及び間隙率 (porosity)、すなわち実際のチップ上に与えられた設計の経路決定の容易さの尺度、の評価が含まれていても良い。但し、これらに限られるわけではない。解析及び続く評価の結果は、通常、所期の目的に用いるために最適なマクロブロックを決定するために用いられる。コンパイル済マクロブロックが解析及び評価されると、プロセス制御は、ステップ604に戻り、次のオプションセットを選択する。プロセス制御がステップ604に戻ると、マクロブロックを用いてすでにコンパイルされたオプションセットに基づいて、所期の目的に用いるために最適なマクロブロックに関する情報が利用可能になる。

【0048】図8は、設計仕様及びHDLテンプレートを用いて合成用HDLを生成するプロセスの詳細を示している。合成用HDLを生成するために用いられるテンプレートが暗号化される。プロセス92は700にて始まり、ステップ702では、テンプレートファイルへのポインタ及びパラメタをテンプレートコンパイラが受け取る。ポインタが指示するテンプレートファイルには、「上位」テンプレート、及び上位テンプレートによって使用される「下位」テンプレートが含まれる。指示されるテンプレートファイルは、暗号化済テンプレートファイルである。ステップ704では、テンプレートファイルは、解読ツール、例えば図4を参照して前述したサードパーティ製の解読ツール、を用いて解読される。一般的に、上位テンプレートファイル、及び上位テンプレートファイルによって要求される下位テンプレートファイルのみが解読される。

【0049】上位テンプレートファイルは、ステップ706にて実行される。この上位テンプレートファイルの実行により、全ての必要なサブファイル、あるいは、上位テンプレートファイルが要求することの可能な従属下位テンプレートファイルが実行される。ある実施形態では、下位テンプレートファイルが、他の下位テンプレートファイルの実行を引き起こしても良い。合成用HDL生成プロセスにおける上位テンプレートファイルの実行ステップ、及び必要なサブファイルの実行ステップは、呼出のネスティングに備える。すなわち、各サブファイルは、実行されるべきサブファイルが無くなるまで継続して実行される。サブファイルは、順次に他のサブファイルを読み出すことができる。当業者であれば理解できるように、ファイル又はサブファイルの実行は、再帰的な実行 (recursive execution) という結果になっても良い。すなわち、上位ファイル、あるいは、サブファイ

ルは、一連の呼出を通じて自身を直接的、あるいは、間接的に呼び出すことができる。本実施形態では、この再帰 (recursion) は、オペレーティングシステムによって処理される。テンプレートファイルの実行の詳細は、図9を参照して後述する。上位テンプレートファイル、及び全ての必要なサブファイルの実行により、合成用HDLの生成およびマクロブロックのネットリストへのコンパイルが行われる。この後、合成用HDLの生成プロセスは708にて完了する。

10 【0050】図9は、テンプレートファイル実行プロセスを詳細に示している。このプロセス706は、800にて始まる。ステップ802では、テンプレートファイルからコマンドが選択される。選択されたコマンドは、一般的に、テンプレートファイル中にて入手可能な第1の非実行コマンドライン、あるいは、次の命令である。上位テンプレートファイルは、コード化されたコマンドをも含む場合があるが、上位テンプレートファイルは、通常、下位テンプレートファイルを読み出すように機能する。上位ファイルの例は、図13に見ることができる。ステップ804では、テンプレートファイル (これは上位テンプレートファイルであっても良い) からの最後のコマンドが実行されたかどうか判定される。実行されるべき命令が残っていない場合には、テンプレートファイル実行プロセスは、810にて完了する。テンプレートファイル実行プロセスは再帰的である。すなわち、最初の上位テンプレートからの全ての命令が実行されるまで、合成用HDLを生成しインスタンス化マクロをコンパイルするプロセスは継続する。すなわち、プロセス706が繰り返し実行されることになる。

30 【0051】実行されるべきコマンドが存在する場合には、プロセス制御は、ステップ806に進む。ステップ806では、どのタイプの命令がテンプレートファイルから選択されたかが判定される。本実施形態では、次の4つのタイプのコマンドを選択することができる。すなわち、マクロブロックプログラムを読み出すコマンド、他のテンプレートファイルを実行するコマンド、上位テンプレートファイル中に含まれるコード化済コマンドを実行するコマンド、及びテンプレートグラフィカルユーザインターフェース (GUI) へ戻るコマンドの四つである。

40 【0052】呼び出されるコマンドのタイプがマクロブロックプログラムを読み出すコマンドである場合、プロセス制御は、ステップ806からステップ808に移行する。ステップ808では、マクロブロックプログラム、あるいは、コンパイラが呼び出される。マクロブロックプログラムは、GUIにて特定されたパラメタに基づいて適当なマクロブロック用のネットリストを作成する。マクロブロックプログラムが呼び出された後、プロセス制御は、ステップ802に戻り、実行されるべきテンプレートファイルから次のコマンドが選択される。

【0053】呼び出されるコマンドのタイプがテンプレートファイル実行の呼出であるとステップ806で判定された場合には、図8を参照して前述したように、ステップ810にて合成用HDL生成ステップ全体が繰り返し実行され、任意の要求暗号化テンプレートファイルが解読される。テンプレートファイルが解読された後、プロセス制御はステップ802に戻る。再帰法がサポートされているので、呼び出されているテンプレートファイルは、呼出元テンプレートと同一であっても良い。

【0054】呼び出されたコマンドのタイプがテンプレートGUIに戻る要求であるとステップ806で判定された場合には、プロセス制御はステップ806からステップ818に進む。ステップ818では、解読ツールが呼び出され、未だ解読されていない任意の要求テンプレートファイルが解読される。換言すれば、図8を参照して前述したように、合成用HDL生成ステップ全体が繰り返し実行され、任意の要求暗号化テンプレートファイルが解読される。例えば、上位テンプレートファイルはUARTに対して固有であるが、FIFOメモリに関連する情報が上位テンプレートファイルによって呼び出される場合、FIFOメモリに関連するテンプレートファイルを解読する必要がある可能性がある。ある実施形態では、GUIを用いることで、論理設計者が「新規」な、依然として暗号化されている、解読されるべきテンプレートファイルに関連する設計仕様を入力することができる。他の実施形態では、論理設計者からのいかなる新規入力をも用いることなく、新規のテンプレートファイルに関連するパラメタファイル中の仕様を用いられる。テンプレートファイルが解読された後、プロセス制御はステップ802に戻る。

【0055】テンプレートファイルが解読され、ネットリストを用いて論理設計がレイアウトされた後は、図2を参照して前述したように、集積回路マスクを生成するためにマスク製造ツールを用いることができる。この集積回路マスクは、集積回路チップまたはダイを生成するために用いられる。このマスク生成装置は、ネットリストを合成するために用いられる機械と同じもの又はその一部であっても良い。一般的に、フォトリソグラフィプロセスでは、マスクを用いて集積回路が生成される。チップ自身が製造されると、ダイ上の集積回路は外部回路に対する接続を備えていなければならない。これは、一般的に、集積回路にボンディングワイヤ及び／またはリードフレームを取り付けることによって達成される。こ

の後、回路は、プラスチック等のパッケージ材料に封入され、これにより、集積回路が製造される。

【0056】以上、いくつかの発明の実施の形態に基づき本発明を説明したが、本発明の趣旨から逸脱しない範囲で種々の変更物、改良物、均等物が存在する。また、本発明に係る方法および装置の双方を実現する他の種々の方法が存在することにも留意すべきである。

【図面の簡単な説明】

【図1】 従来技術に係る集積回路の設計及び製造プロセスを示すフローチャートである。

【図2】 本発明に係る集積回路の設計及び製造プロセスを示すフローチャートである。

【図3】 集積回路の形成に用いられるシステムを示すブロック図である。

【図4】 HDLテンプレート生成プロセスを詳細に示すフローチャートである。

【図5】 設計仕様開発プロセスを詳細に示すフローチャートである。

【図6】 パラメタファイル中に備えられている情報に基づき生成されるメニュー表示を示す説明図である。

【図7】 最良マクロブロックの決定プロセスを詳細に示すフローチャートである。

【図8】 合成用HDLを生成するために設計仕様及びHDLテンプレートを使用するプロセスを詳細に示すフローチャートである。

【図9】 テンプレートファイル実行プロセスを詳細に示すフローチャートである。

【図10】 パラメタファイル及びパラメタチェックファイルの第1の例を示す説明図である。

【図11】 パラメタファイル及びパラメタチェックファイルの第2の例を示す説明図の前半部分である。

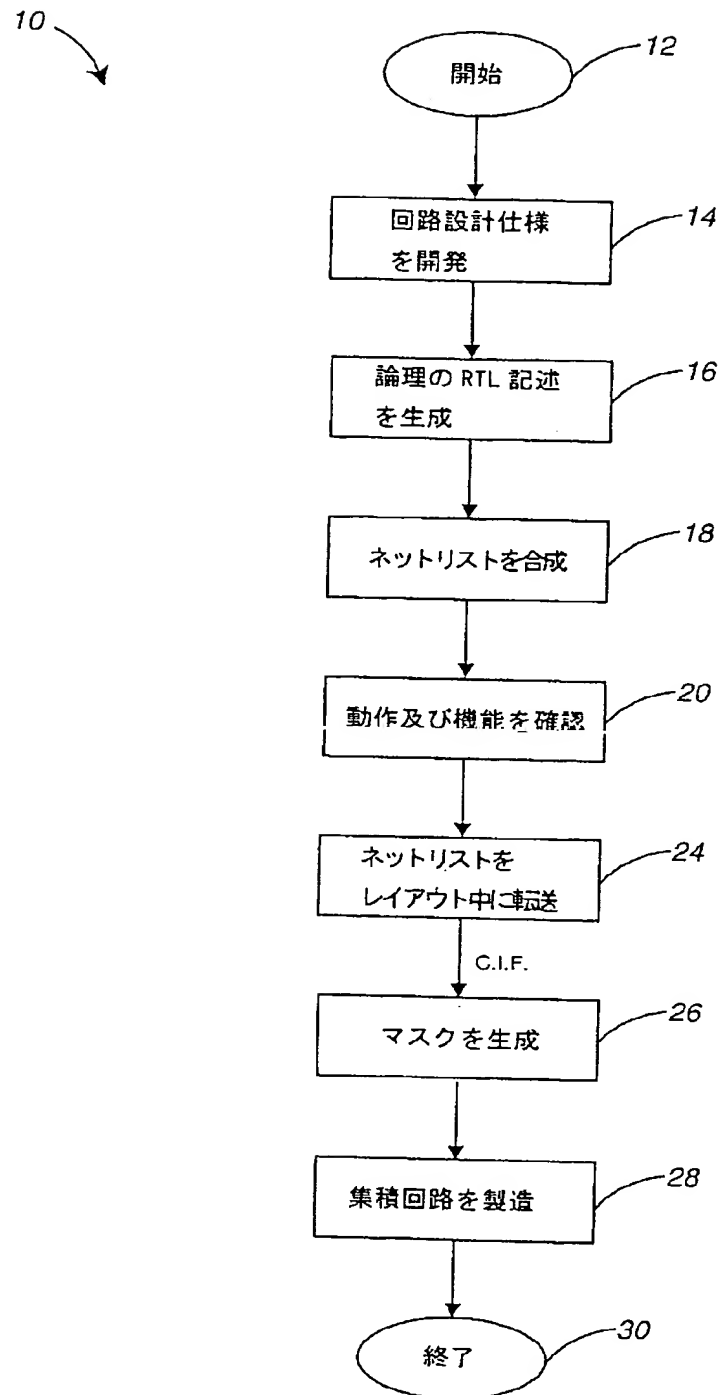
【図12】 パラメタファイル及びパラメタチェックファイルの第2の例を示す説明図の後半部分である。

【図13】 上位ファイルを例示する説明図である。

【符号の説明】

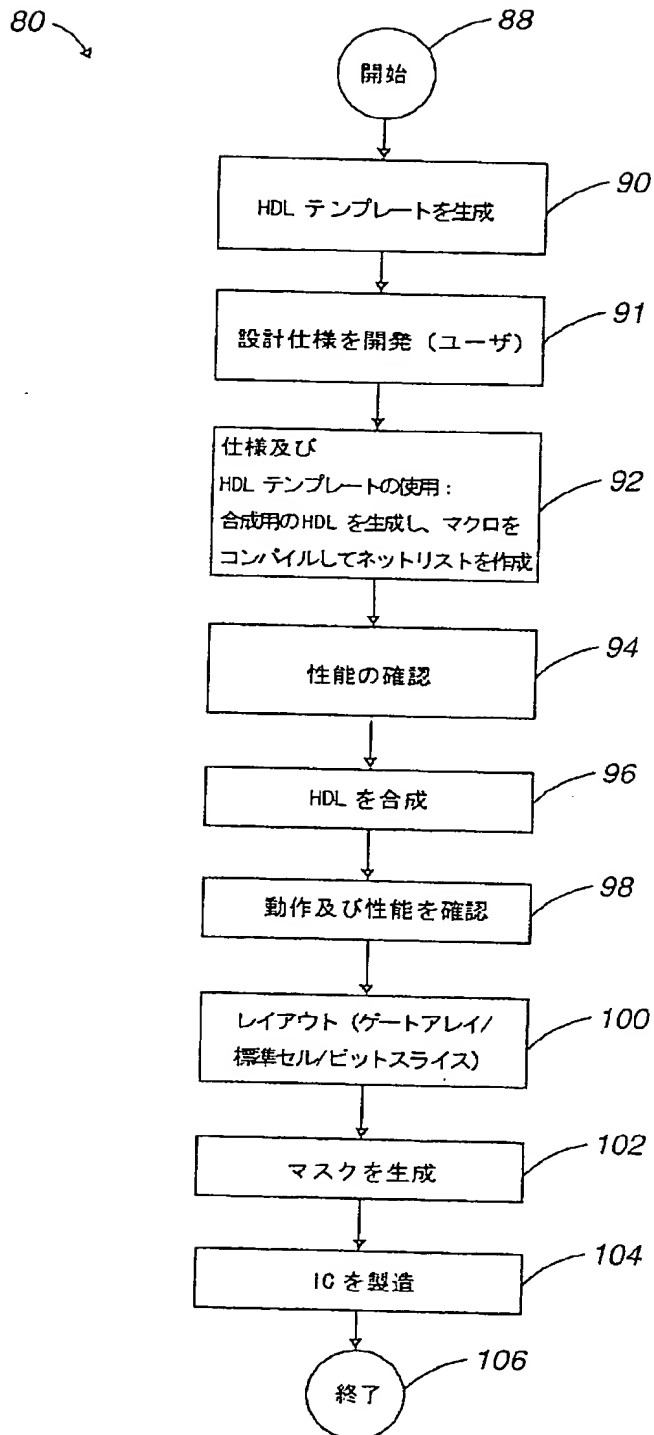
31…集積回路製造システム、32…中央演算装置(CPU)、34…I/Oスロット、36…キーボード、38…モニタ、40…ROM、42…RAM、44…ディスクドライブ、46…マスク生成装置、48…IC製造装置、50…パッケージIC、52…ダイ、53…リード、500…メニュー、504…ワード実行、506…ワード幅。

【図1】

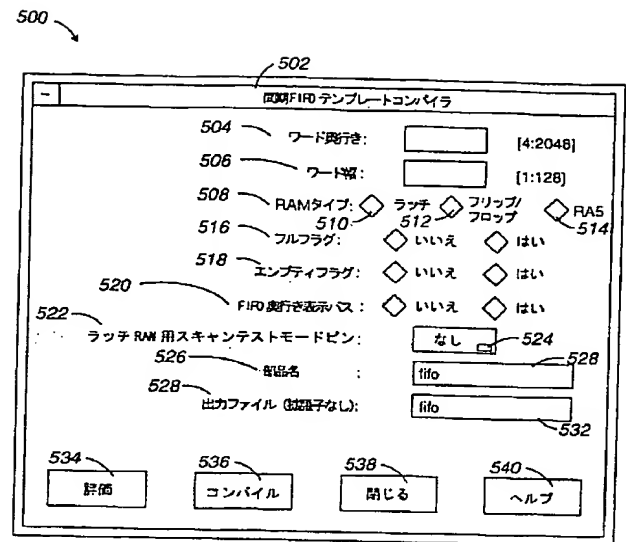


従来技術

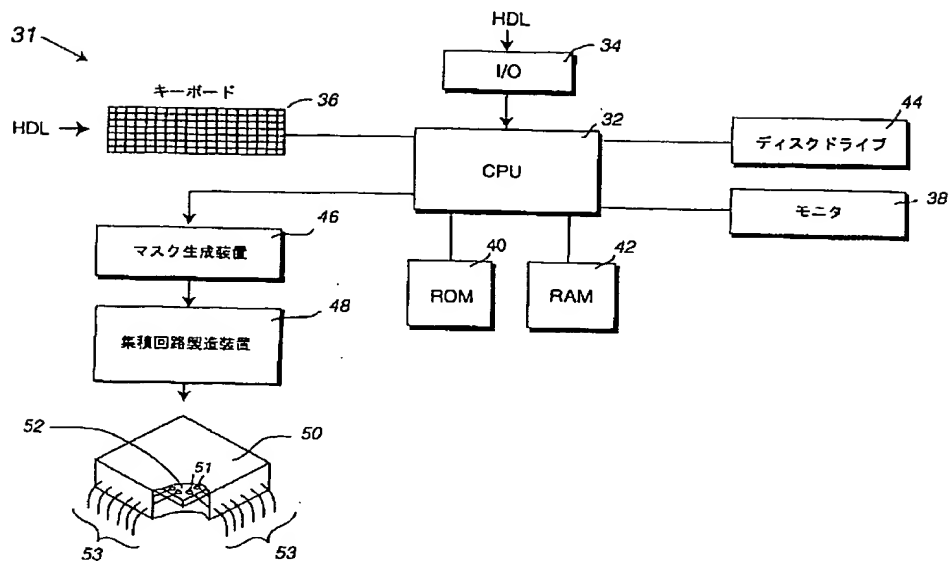
【図2】



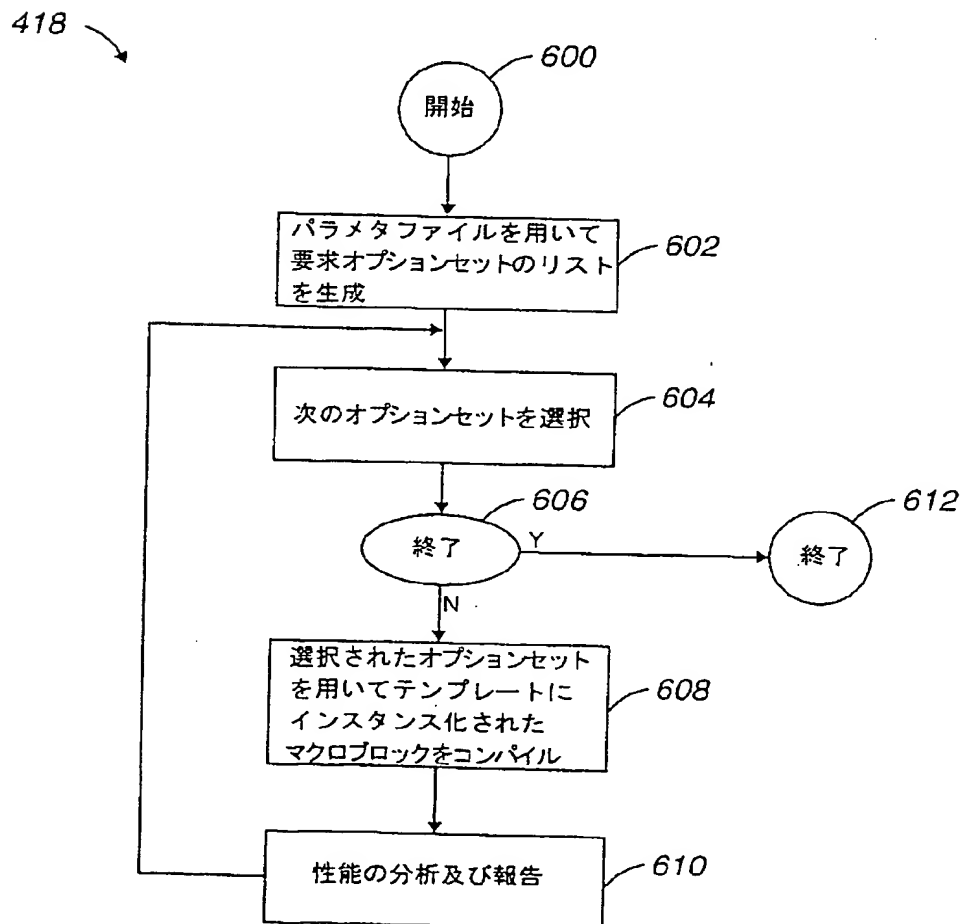
【図6】



【図3】

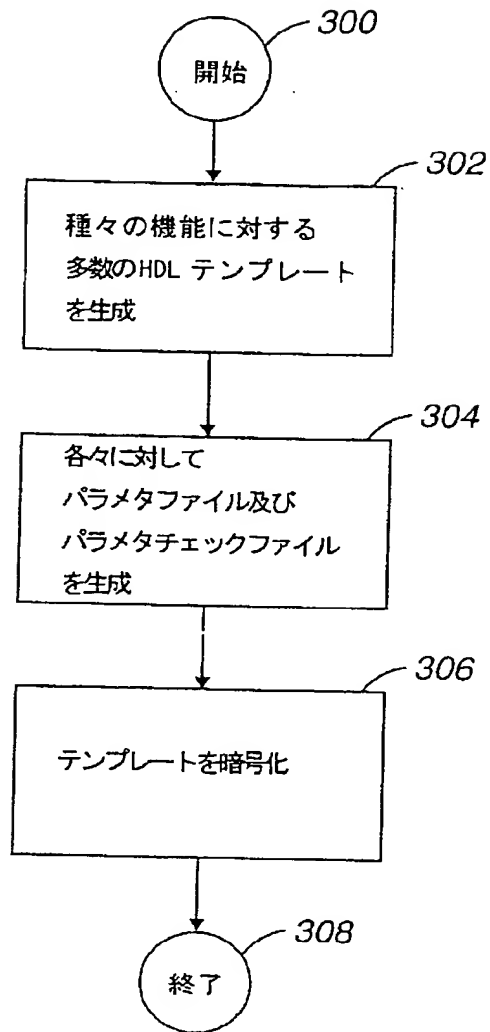


【図7】



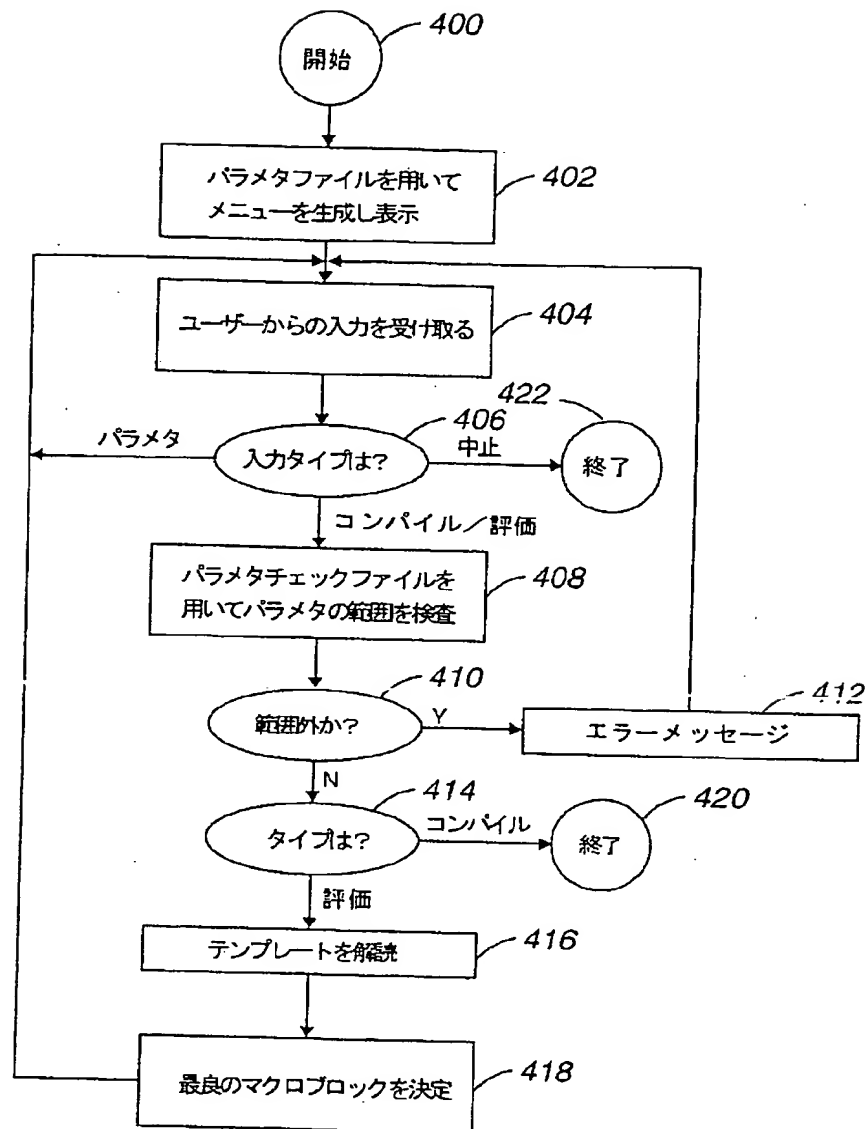
【図4】

90



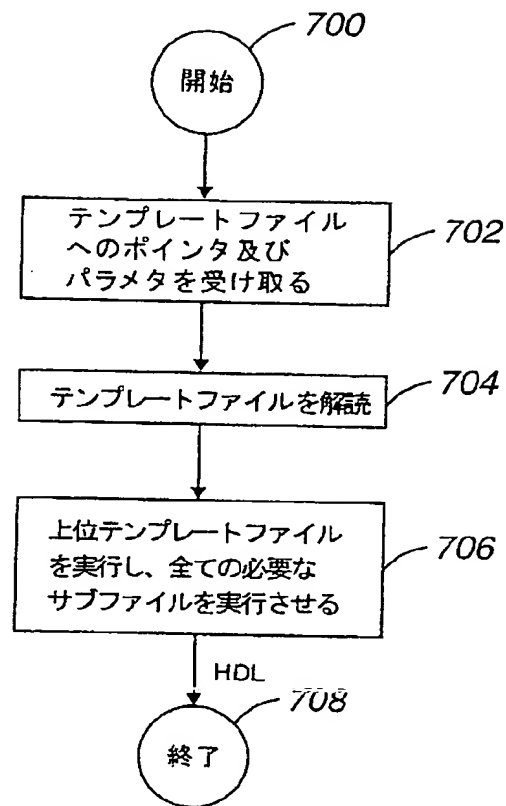
【図5】

91 ↗

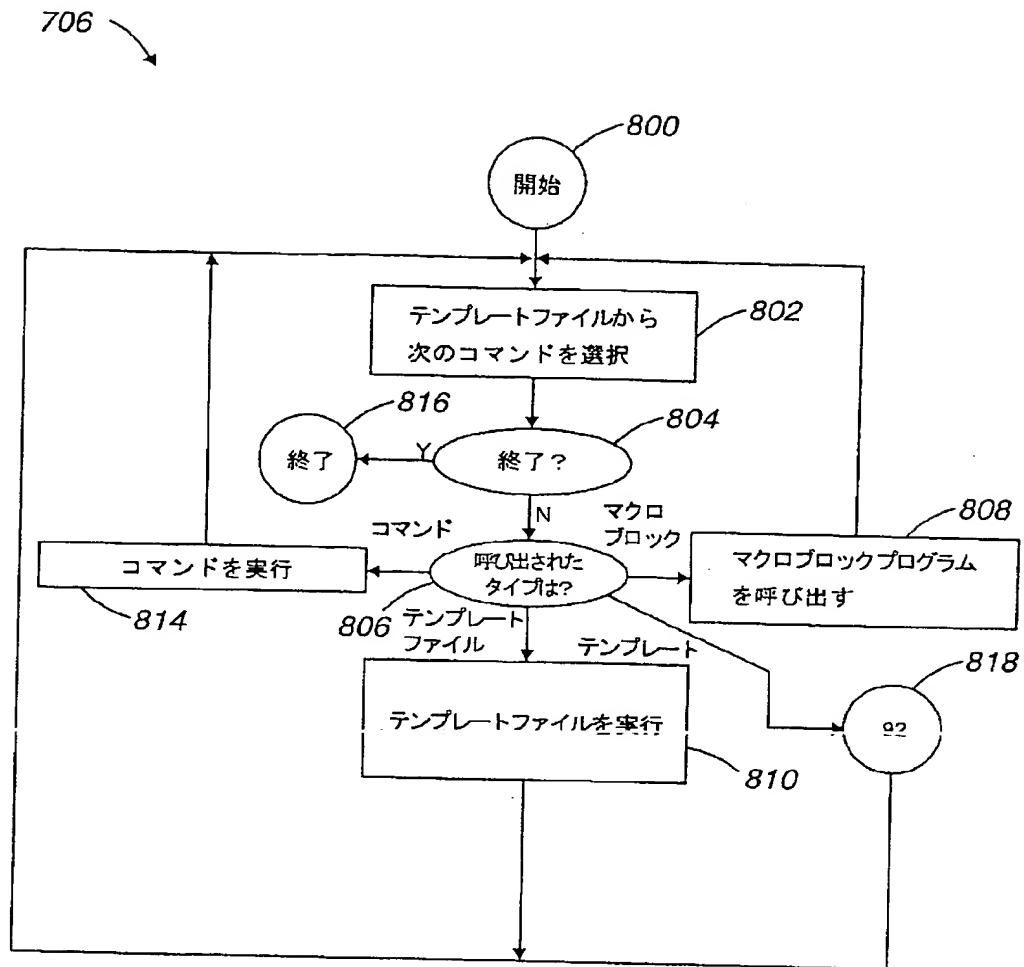


【図8】

92



【図9】



【図10】

```

TITLE      "Synchronous Interface FIFO - 1 Clock - Template"
MENUTITLE  "Synchronous Interface FIFO - 1 Clock"
CHECKFILE  "fifo.hpc"
VERILOGHDLFILE "fifo_ver.htf"
VHDLHDLFILE "fifo_vhd.htf"
HELPPFILE  "sisfifo.pdf"
SHORTNAME  "sisfifo"
VERSION    "1.0"
TEMPLATEENABLED

```

DESCRIPTION -

"This is a fifo template compiler, with synchronous read and write \n-
port interfaces and a single clock for read and write port operations.\n-
Providing Full, Empty, and Depth indications. \n-
\n-

Example:\n-

To compile a 8x8 fifo with Full & Empty flags\n-
hdlc -shell sisfifo -FE -d 8 -b 8 -o test"

#	variable	GUI	Shell	Field	Right Hand	Elements
#	name	question	tag	type	Text	
TP	DEPTH	"Word Depth:"	-d	INTEGER	"[4:2048]"	"Word depth"
TP	WIDTH	"Word Width:"	-b	INTEGER	"[1:128]"	"Word width"
TP	RAMTYPE	"Memory Type:"	-t	RADIO		
		"Latch" "Flip/Flop" "RAS" -				
		"Use Latches for the memory core." -				
		"Use Flip/Flops for the memory core." -				
		"Use a compiler RAS ram for the memory core."				
TP	SCAN	"Scan Testmode pin for Latched Memory:" -T		OPTION	"None"	
		"Buffered" "Unbuffered" "No Testmode provided" "Buffered Testmode pin provided" "Unbuffered Testmode pin provided"				
TP	DECODE	"Read Decode Type for Latched Memory:" -D		RADIO	0	
		"Muxed" "Tristate" -				
		"Use Muxes to generate the Read decode" "Use Tristates to generate the Read decode"				
TP	RESET	"Controller Reset Type:" -r		RADIO	0	"Asynchronous"
		"Synchronous" -				
		"Provide a asynchronous FIFO controller reset" "Provide a synchronous FIFO controller reset"				
TP	FULL	"Full Flag?"	+F	RADIO		"No" "Yes" -
		"Provide full flag"				
TP	EMPTY	"Empty Flag?"	+E	RADIO		"No" "Yes" -
		"Provide empty flag"				
TP	LEVELBUS	"FIFO Depth Indicator Bus?" +L		RADIO		"No"
		"Yes" "Provide Depth bus"				

【図 1 1】

```
#print("Synchronous Interface - Synchronous FIFO - Version ",VERSION,"\\n");

#if (DEPTH < 4)
# error("The \\\"Word Depth\\\" must be greater than 4.\\n");
# exit(1);
#endif
##
#if (DEPTH > 2048)
# error("The \\\"Word Depth\\\" must be less than or equal to 2048.\\n");
# exit(1);
#endif
##
#if ( DEPTH*WIDTH > 16384 && RAMTYPE == 2)
# error("The total number of bits for a RA5 RAM must be less than 16384.\\n");
# exit(1);
#endif
##
#if ( DEPTH > 64 && RAMTYPE < 2)
# error("If the depth is greater than 64 the \\\"RAM Type\\\" must be of a compiled type
(ie. RA5).\\n");
# exit(1);
#endif
##
#if (WIDTH < 1)
# error("The \\\"Word Width\\\" must be greater than 0.\\n");
# exit(1);
#endif
##
#if (WIDTH > 128)
# error("The \\\"Word Width\\\" must be less than or equal to 128.\\n");
# exit(1);
#endif
##
#if (SCAN && RAMTYPE == 1)
# error("A Flip/Flop memory is fully SCAN testable, so this options is not
needed.\\n");
# exit(1);
#endif
##
#if (SCAN && RAMTYPE > 1)
# error("The Scan option is only available for Latch based memories.\\n");
# exit(1);
#endif
##
```

【図 1 2】

```
#if (RAMTYPE == 2 && (substr(TARGETLIB,1,4) != "vsc7" &&
substr(TARGETLIB,1,4) != "vsc6"))
# error("The RA5 Ram is only supported in cell-based .6um and .5um
technologies.\n");
# exit(1);
#endif
##
#if (RAMTYPE == 2 && DEPTH < 8)
# error("The RA5 Ram only supports depths greater than 8.\n");
# exit(1);
#endif
##
#if (ESTONLY)
All options are valid for this template. Push the "View DOC" button for
more information.
#endif
#
#include fifo_conns.hf
#if (LANGUAGE & LANG_VERILOG)
#include fifo_ver_doc.hf
#endif
#
#if (LANGUAGE & LANG_VHDL)
#include fifo_vhd_doc.hf
#endif
#
#endif ;
```

【図 1 3】

```
#string RAM_NAME;
#string DLY_NAME;
#int status;
#int statusall;
#status = 0;
#statusall = 0;
#RAM_NAME = FILENAME "_ram";
#DLY_NAME = FILENAME "_dly";
#
## BUILD THE PORT LIST
#include fifo_ver_head.hf
#
#if (RAMTYPE == 0)
## LATCH RAM
#include fifo_ver_0.hf
#endif
#
#if (RAMTYPE == 1)
## FF RAM
#include fifo_ver_1.hf
#endif
#
#if (RAMTYPE == 2)
## RA5 RAM
#include fifo_ver_2.hf
#endif
#
## BUILD THE TESTBENCH
#include fifo_ver_tb.hf
#
#if (TOOLSET == "Synopsys")
## BUILD THE SYNOPSYS SYNTHESIS SCRIPT
#include fifo_ver_syn.hf
#endif
#
#if (TOOLSET == "Compass")
## BUILD THE COMPASS SYNTHESIS SCRIPT
#include fifo_ver_cmp.hf
#endif
#
#exit(statusall);
```